

Usando cookies HTTP

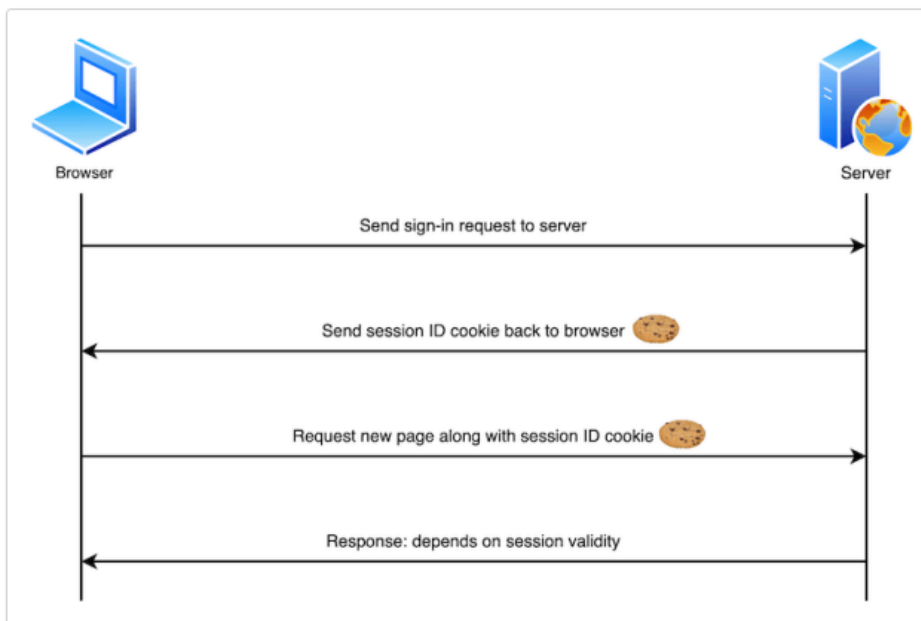
Um cookie (também conhecido como cookie da web ou cookie de navegador) é um pequeno pedaço de dado que um servidor envia ao navegador da web de um usuário. O navegador pode armazenar cookies, criar novos cookies, modificar cookies existentes e enviá-los de volta ao mesmo servidor em requisições posteriores. Cookies permitem que aplicações web armazenem quantidades limitadas de dados e lembrem informações de estado; por padrão, o protocolo HTTP é sem estado.

Neste artigo, exploraremos os principais usos de cookies, explicaremos as melhores práticas para usá-los e examinaremos suas implicações de privacidade e segurança.

Para que os cookies são usados

Normalmente, o servidor usará o conteúdo dos cookies HTTP para determinar se diferentes requisições vêm do mesmo navegador/usuário e então emitir uma resposta personalizada ou genérica conforme apropriado. A seguir, descreve-se um sistema básico de login de usuário:

1. O usuário envia credenciais de login para o servidor, por exemplo, através de um envio de formulário.
2. Se as credenciais estiverem corretas, o servidor atualiza a interface de usuário para indicar que o usuário está logado e responde com um cookie contendo um ID de sessão que registra o status de login no navegador.
3. Em um momento posterior, o usuário navega para uma página diferente no mesmo site. O navegador envia o cookie contendo o ID de sessão junto com a requisição correspondente para indicar que ainda considera o usuário logado.
4. O servidor verifica o ID de sessão e, se ainda for válido, envia ao usuário uma versão personalizada da nova página. Se não for válido, o ID de sessão é deletado e o usuário vê uma versão genérica da página (ou talvez veja uma mensagem de "acesso negado" e seja solicitado a fazer login novamente).



Cookies são usados principalmente para três propósitos:

- **Gerenciamento de sessão:** status de login do usuário, conteúdo do carrinho de compras, pontuações de jogos ou quaisquer outros detalhes relacionados à sessão do usuário que o servidor precisa lembrar.
 - **Personalização:** preferências do usuário, como idioma de exibição e tema da interface.
 - **Rastreamento:** registrar e analisar o comportamento do usuário.
-

Armazenamento de dados

Nos primeiros dias da web, quando não havia outra opção, cookies eram usados para propósitos gerais de armazenamento de dados no lado do cliente. APIs modernas de armazenamento são agora recomendadas, por exemplo, a Web Storage API (localStorage e sessionStorage) e IndexedDB.

Essas APIs são projetadas para armazenamento, nunca enviam dados ao servidor e não apresentam outras desvantagens de usar cookies para armazenamento:

- Os navegadores geralmente limitam o número máximo de cookies por domínio (varia por navegador, geralmente nas centenas), e um tamanho máximo por cookie (normalmente 4KB). As APIs de armazenamento podem armazenar quantidades maiores de dados.
- Cookies são enviados com cada requisição, o que pode piorar o desempenho (por exemplo, em conexões móveis lentas), especialmente se você tiver muitos cookies definidos.

Criando, removendo e atualizando cookies

Após receber uma requisição HTTP, um servidor pode enviar um ou mais cabeçalhos Set-Cookie com a resposta, cada um dos quais definirá um cookie separado. Um cookie é definido especificando um par nome-valor como este:

```
http                                                                    Copiar  Editar

Set-Cookie: <nome-do-cookie>=<valor-do-cookie>
```

A seguinte resposta HTTP instrui o navegador a armazenar um par de cookies:

```
http                                                                    Copiar  Editar

HTTP/2.0 200 OK
Content-Type: text/html
Set-Cookie: yummy_cookie=chocolate
Set-Cookie: tasty_cookie=strawberry
[conteúdo da página]
```

Quando uma nova requisição é feita, o navegador normalmente envia os cookies previamente armazenados para o domínio atual de volta ao servidor dentro de um cabeçalho HTTP Cookie:

```
h                                                                    Copiar  Editar

GET /sample_page.html HTTP/2.0
Host: www.example.org
Cookie: yummy_cookie=chocolate; tasty_cookie=strawberry
```

Remoção: definindo a duração de vida de um cookie

Você pode especificar uma data de expiração ou período de tempo após o qual o cookie deve ser deletado e não mais enviado. Dependendo dos atributos definidos no cabeçalho Set-Cookie quando os cookies são criados, eles podem ser cookies permanentes ou de sessão:

- **Cookies permanentes** são deletados após a data especificada no atributo Expires:

```
http                                                                    Copiar  Editar

Set-Cookie: id=a3fWa; Expires=Thu, 31 Oct 2021 07:28:00 GMT;
```

- Ou após o período especificado no atributo Max-Age:

```
h
Set-Cookie: id=a3fWa; Max-Age=2592000
```

- **Cookies de sessão** — cookies sem atributos Max-Age ou Expires — são deletados quando a sessão atual termina. O navegador define quando a "sessão atual" termina, e alguns navegadores usam restauração de sessão ao reiniciar. Isso pode fazer com que cookies de sessão durem indefinidamente.

Nota: Expires está disponível há mais tempo que Max-Age, contudo Max-Age é menos propenso a erros e tem precedência quando ambos são definidos.

Nota: Se seu site autentica usuários, ele deve regenerar e reenviar cookies de sessão, mesmo os que já existem, sempre que um usuário autenticar. Essa abordagem ajuda a prevenir ataques de fixação de sessão.

Técnicas para recriar cookies deletados

Existem algumas técnicas projetadas para recriar cookies após serem deletados. Essas são conhecidas como **cookies "zumbis"**. Essas técnicas violam os princípios de privacidade e controle do usuário, podem violar regulamentos de privacidade de dados e podem expor um site que as utilize a responsabilidade legal.

Atualizando valores de cookies

Para atualizar um cookie via HTTP, o servidor pode enviar um cabeçalho Set-Cookie com o mesmo nome do cookie existente e um novo valor. Por exemplo:

```
http
Set-Cookie: id=new-value
```

Existem várias razões pelas quais você pode querer fazer isso, por exemplo, se um usuário atualizou suas preferências e a aplicação deseja refletir as alterações nos dados do lado do cliente (você também pode fazer isso com um mecanismo de armazenamento no lado do cliente como o Web Storage).

Atualizando cookies via JavaScript

No navegador, você pode criar novos cookies via JavaScript usando a propriedade `document.cookie`, ou a API assíncrona **Cookie Store API**. Observe que todos os exemplos abaixo usam `document.cookie`, pois é a opção mais amplamente suportada/estabelecida.

Você também pode acessar cookies existentes e definir novos valores para eles, desde que o atributo `HttpOnly` **não** esteja definido neles (isto é, no cabeçalho Set-Cookie que os criou):

```
javascript
document.cookie = "yummy_cookie=chocolate";
document.cookie = "tasty_cookie=strawberry";
console.log(document.cookie);
// logs "yummy_cookie=chocolate; tasty_cookie=strawberry"

document.cookie = "yummy_cookie=blueberry";
console.log(document.cookie);
// logs "tasty_cookie=strawberry; yummy_cookie=blueberry"
```

Nota: Para fins de segurança, você **não pode** alterar valores de cookies enviando diretamente um cabeçalho Cookie atualizado ao iniciar uma requisição, ou seja, via `fetch()` ou `XMLHttpRequest`.

Também há boas razões para **não permitir** que o JavaScript modifique cookies — ou seja, defina `HttpOnly` durante a criação. Veja a seção de **Segurança** para mais detalhes.

Segurança

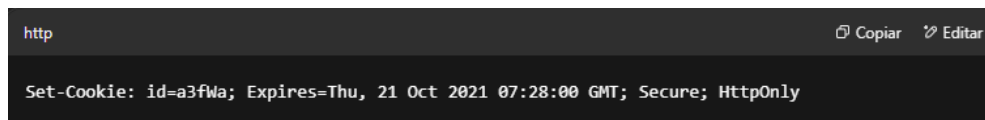
Quando você armazena informações em cookies, por padrão todos os valores de cookies são visíveis para o usuário final e podem ser alterados por ele. Você realmente não quer que seus cookies sejam mal utilizados — por exemplo, acessados/modificados por agentes maliciosos ou enviados para domínios onde **não** deveriam ser enviados. As consequências potenciais podem variar de:

- Irritante — aplicativos não funcionando ou apresentando comportamento estranho
- Até catastrófico — um criminoso poderia, por exemplo, roubar um ID de sessão e usá-lo para definir um cookie que faz parecer que ele está logado como outra pessoa, tomando o controle da conta bancária ou de e-commerce da vítima.

Você pode proteger seus cookies de várias formas, descritas nesta seção.

Bloqueando acesso aos seus cookies

Você pode garantir que os cookies sejam enviados com segurança e não sejam acessados por partes ou scripts não intencionados de duas maneiras: com os atributos `Secure` e `HttpOnly`:



```
http
Set-Cookie: id=a3fWq; Expires=Thu, 21 Oct 2021 07:28:00 GMT; Secure; HttpOnly
```

- Um cookie com o atributo `Secure` **somente** é enviado ao servidor com uma requisição criptografada sobre o protocolo HTTPS. Nunca é enviado com HTTP não seguro (exceto em localhost), o que significa que atacantes man-in-the-middle não podem acessá-lo facilmente. Sites inseguros (com `http:` na URL) **não** podem definir cookies com o atributo `Secure`. Contudo, **não presume** que `Secure` impede todo acesso à informação sensível em cookies. Por exemplo, alguém com acesso ao disco rígido do cliente (ou JavaScript se o atributo `HttpOnly` **não** estiver definido) pode ler e modificar as informações.
- Um cookie com o atributo `HttpOnly` **não pode ser acessado por JavaScript**, por exemplo usando `document.cookie`; ele só pode ser acessado quando chega ao servidor. Cookies que mantêm sessões de usuário, por exemplo, **devem** ter o atributo `HttpOnly` definido — seria muito inseguro torná-los disponíveis ao JavaScript. Esta precaução ajuda a mitigar ataques de Cross-site Scripting (XSS).

Definindo onde os cookies são enviados

Os atributos `Domain` e `Path` definem o escopo de um cookie: para quais URLs os cookies são enviados.

- O atributo `Domain` especifica qual servidor pode receber um cookie. Se especificado, os cookies ficam disponíveis no servidor indicado **e em seus subdomínios**. Por exemplo, se você definir `Domain=mozilla.org` a partir de `mozilla.org`, os cookies estarão disponíveis nesse domínio e em subdomínios como `developer.mozilla.org`.
- Se o cabeçalho `Set-Cookie` **não** especificar um atributo `Domain`, os cookies estarão disponíveis **apenas no servidor que os definiu**, e **não** em seus subdomínios.

Portanto, especificar Domain é menos restritivo do que omiti-lo. Observe que um servidor só pode definir o atributo Domain para **seu próprio domínio** ou um domínio pai, **não** para um subdomínio ou outro domínio.

Exemplo: um servidor com domínio foo.example.com pode definir o atributo como example.com ou foo.example.com, mas **não** como bar.foo.example.com ou elsewhere.com. (Os cookies ainda serão enviados para subdomínios como bar.foo.example.com, no entanto.)

Veja a seção *Domínios inválidos* para mais detalhes.

- O atributo Path indica um caminho de URL que deve existir na URL requisitada para que o cabeçalho Cookie seja enviado. Por exemplo:

```
http
Set-Cookie: id=a3fWa; Expires=Thu, 21 Oct 2021 07:28:00 GMT; Secure; HttpOnly; Path=/docs
```

O caractere %x2F ("/") é considerado um separador de diretório, e subdiretórios também correspondem. Por exemplo, se você definir Path=/docs, estes caminhos de requisição **correspondem**:

```
bash
/docs
/docs/
/docs/Web/
/docs/Web/HTTP
```

Mas **estes não correspondem**:

```
bash
/
/docsets
/fr/docs
```

Nota: o atributo path permite controlar quais cookies o navegador envia com base nas diferentes partes de um site. Não é destinado como uma medida de segurança e **não protege** contra a leitura não autorizada do cookie a partir de um caminho diferente.

Controlando cookies de terceiros com SameSite

O atributo SameSite permite que os servidores especifiquem **se/quando** os cookies são enviados com requisições entre sites — ou seja, cookies de terceiros.

Requisições entre sites são aquelas em que o site (o domínio registrável) e/ou o esquema (http ou https) **não coincidem** com o site que o usuário está visitando atualmente.

Isso inclui requisições enviadas quando links são clicados em outros sites para navegar até o seu site, ou qualquer requisição enviada por conteúdo de terceiros incorporado.

SameSite ajuda a prevenir o vazamento de informações, preservando a privacidade do usuário e oferecendo alguma proteção contra ataques de falsificação de requisição entre sites (CSRF). Ele aceita três valores possíveis:

- **Strict:** faz com que o navegador envie o cookie **somente** em resposta a requisições originadas do mesmo site de origem do cookie.
Deve ser usado quando você tem cookies relacionados a funcionalidades que sempre estarão por trás de uma navegação inicial, como autenticação ou armazenamento de informações do carrinho de compras.

```
http
Set-Cookie: cart=110045_77895_53420; SameSite=Strict
```

- **Lax:** é similar, exceto que o navegador também envia o cookie quando o usuário **navega até o site de origem do cookie** (mesmo vindo de um site diferente).

Isso é útil para cookies que afetam a exibição de um site — por exemplo, você pode ter informações de produtos parceiros junto com um link de afiliado no seu site.

Quando esse link é seguido até o site parceiro, ele pode querer definir um cookie indicando que o link de afiliado foi seguido, o que exibe um banner de recompensa e fornece desconto se o produto for comprado.

```
http
Set-Cookie: affiliate=e4rt45dw; SameSite=Lax
```

- **None:** especifica que cookies são enviados em **requisições de origem e entre sites**. Isso é útil se você quiser enviar cookies junto com requisições feitas por conteúdo de terceiros incorporado em outros sites — por exemplo, provedores de tecnologia de anúncios ou de análise (analytics). Observe que se SameSite=None for definido, o atributo Secure **também deve** ser definido — SameSite=None requer um contexto seguro.

```
http
Set-Cookie: widget_session=7yjgj57e4n3d; SameSite=None; Secure; HttpOnly
```

Nota: Cookies usados para informações sensíveis também devem ter um tempo de vida curto.

Nota: Se nenhum atributo SameSite for definido, o cookie será tratado como Lax por padrão.

Prefixos de cookies

Devido ao design do mecanismo de cookies, um servidor **não pode confirmar** que um cookie foi definido a partir de uma origem segura ou mesmo dizer onde um cookie foi originalmente definido.

Uma aplicação em um subdomínio pode definir um cookie com o atributo Domain, o que dá acesso a esse cookie em **todos os outros subdomínios**. Esse mecanismo pode ser abusado em um ataque de **fixação de sessão** (*session fixation*).

Como uma **medida de defesa em profundidade**, você pode usar prefixos de cookies para declarar fatos específicos sobre o cookie. Dois prefixos estão disponíveis:

- **__Host-:** se o nome de um cookie tem esse prefixo, ele **só é aceito** em um cabeçalho Set-Cookie **se também** estiver marcado com o atributo Secure, **for enviado de uma origem segura, não incluir** um atributo Domain **e tiver** o atributo Path definido como /. Em outras palavras, o cookie é bloqueado ao domínio (domain-locked).
- **__Secure-:** se o nome de um cookie tem esse prefixo, ele **só é aceito** em um cabeçalho Set-Cookie **se estiver marcado com** o atributo Secure **e tiver sido enviado de uma origem segura**. Isso é mais fraco que o prefixo __Host-.

O navegador rejeitará cookies com esses prefixos que **não cumprirem suas restrições**.

Isso garante que cookies criados por subdomínios com prefixos sejam **confinados a um subdomínio** ou **totalmente ignorados**. Como o servidor da aplicação só verifica um nome específico de cookie ao determinar se o usuário está autenticado ou se um token CSRF está correto, isso funciona como uma medida de defesa contra **ataques de fixação de sessão**.

Nota: No servidor, a aplicação web deve verificar o **nome completo do cookie**, incluindo o prefixo.

Os agentes de usuário (navegadores) **não removem o prefixo** do cookie antes de enviá-lo no cabeçalho Cookie da requisição.

Privacidade e rastreamento

Anteriormente falamos sobre como o atributo SameSite pode ser usado para controlar quando cookies de terceiros são enviados, e que isso pode ajudar a preservar a privacidade do usuário.

Privacidade é uma consideração muito importante ao construir sites, pois, quando bem feita, pode construir **confiança com os usuários**. Quando mal feita, pode **destruir completamente essa confiança** e causar vários outros problemas.

Cookies de terceiros podem ser definidos por conteúdo de terceiros incorporado em sites via <iframe>s. Eles têm muitos usos legítimos, como:

- Compartilhar informações de perfil do usuário
- Contar impressões de anúncios
- Coletar análises (*analytics*) entre diferentes domínios relacionados

No entanto, cookies de terceiros também podem ser usados para criar experiências invasivas e "assustadoras" para o usuário.

Um servidor de terceiros pode criar um **perfil do histórico de navegação e hábitos de um usuário** com base em cookies enviados para ele pelo mesmo navegador ao acessar múltiplos sites.

O exemplo clássico é quando você pesquisa um produto em um site e depois é perseguido pela internet por anúncios de produtos semelhantes aonde quer que vá.

Os fornecedores de navegadores sabem que os usuários **não gostam** desse comportamento e, como resultado, todos começaram a **bloquear cookies de terceiros por padrão**, ou pelo menos elaboraram planos para seguir nessa direção.

Cookies de terceiros (ou apenas cookies de rastreamento) também podem ser bloqueados por **configurações do navegador** ou **extensões**.

Veja nosso artigo sobre [cookies de terceiros](#) para informações detalhadas sobre cookies de terceiros, os problemas associados a eles e quais alternativas estão disponíveis.

Consulte nossa [página principal de privacidade](#) para mais informações sobre privacidade em geral.

Regulamentações relacionadas a cookies

Nota: O bloqueio de cookies pode fazer com que alguns componentes de terceiros (como widgets de redes sociais) **não funcionem como o esperado**.

À medida que os navegadores impõem mais restrições a cookies de terceiros, os desenvolvedores devem começar a buscar maneiras de **reduzir a dependência deles**.

Legislações ou regulamentações que abrangem o uso de cookies incluem:

- O Regulamento Geral sobre a Proteção de Dados (**GDPR**) na União Europeia
- A Diretriz de Privacidade Eletrônica (**ePrivacy Directive**) na UE
- A Lei de Privacidade do Consumidor da Califórnia (**California Consumer Privacy Act**)

Essas regulamentações têm alcance global.

Elas se aplicam a **qualquer site da web** que seja acessado por usuários dessas jurisdições (UE e Califórnia, com a ressalva de que a lei da Califórnia se aplica apenas a entidades com receita bruta superior a 25 milhões de dólares, entre outras condições).

Essas regulamentações incluem requisitos como:

- Notificar os usuários de que seu site usa cookies
- Permitir que os usuários optem por não receber alguns ou todos os cookies

- Permitir que os usuários usem a maior parte do seu serviço **sem receber cookies**

Pode haver outras regulamentações que regem o uso de cookies em sua localidade.

A responsabilidade é sua de **conhecer e cumprir** essas regulamentações. Existem empresas que oferecem códigos de "banner de cookies" para ajudá-lo a cumprir tais exigências.

Nota: Empresas devem divulgar os **tipos de cookies** que usam em seus sites para fins de transparência e para cumprir regulamentações.

Por exemplo, veja o [aviso do Google](#) sobre os tipos de cookies que utiliza, e o [Aviso de Privacidade da Mozilla](#) para websites, comunicações e cookies.